Preferences in Data Production Planning

Keith Golden and **Ronen Brafman**[*] and **Wanlin Pang**[†]
NASA Ames Research Center
Moffett Field, CA 94035
keith.golden@nasa.gov

Abstract

This paper discusses the data production problem, which consists of transforming a set of (initial) input data into a set of (goal) output data. There are typically many choices among input data and processing algorithms, each leading to significantly different end products. To discriminate among these choices, the planner supports an input language that provides a number of constructs for specifying user preferences over data (and plan) properties. We discuss these preference constructs, how we handle them to guide search, and additional challenges in the area of preference management that this important application domain offers.

1 Introduction

Petabytes of remote sensing data are available from Earth-observing satellites to help measure, understand, and forecast changes in the Earth system, but using these data effectively can be surprisingly hard. The volume and variety of data files and formats are daunting. Simple data management activities, such as locating and transferring files, changing file formats, gridding point data, and scaling and reprojecting gridded data, can consume far more personnel time and resources than the actual data analysis. We address this problem by developing a planner-based agent for data production, called IMAGEbot, that takes data product requests as high-level goals and executes the commands needed to produce the requested data products.

The data production problem consists of converting an initial set of low-level data products into higher-level data products that can be used for science or decision support. The data products we are concerned with are geospatial data measuring specific *variables* of the Earth system, such as precipitation, but our approach is also applicable to other types of data. Higher level data products may be transformed versions of lower level data products, or they may be entirely new products providing estimates or predictions of unknown Earth system variables, such as soil moisture, based on known variables, such as precipitation. These variables are estimated by running one or more computational *models*, such as simulation codes. The models can be precisely characterized in terms of their input and output requirements, which makes them straightforward to represent in an AI system. The models are also generally scale invariant, and thus insensitive to details of their gridded data inputs, such as resolution and projection, as long as the inputs are coregistered.[1]

Script-based approaches to automation specialize in particular data formats and resolutions in order to simplify the programming. To get the model inputs into a common format acceptable to the model, it is generally necessary to apply various data transforms, such as reprojection of grid data into a common projection, mosaicking or subsetting to change the spatial extent and conversion of point data to grid data. Further processing is needed to visualize the results, such as generation of false-color images, trend graphs and histograms.

Our IMAGEbot system goes beyond the use of scripts to provide for a much more adaptive and flexible planning-based approach. IMAGEbot views data production as a planning problem whose initial state describes the current set of available (typically, low-level) data products, and whose goal state describes the properties of the desired high-level data product. Operators correspond to data transformation and generation tools. Thus, IMAGEbot can be viewed as a tool for automatically generating plans (or scripts) for a particular need. This is a much more powerful approach: IMAGEbot is not tied to a specific set of input formats or a specitic end product, it can handle diverse end-products and initial data sources.

However, the data production problem differs from standard planing problems in a number of aspects, and in this paper we wish to focus on its goal specification. In the data production domain, for any given Earth system variable, there are generally several data products to choose from, which differ along a number of dimensions, such as spatial and temporal resolution, spatial and temporal coverage and quality. Models and other data transforms also vary along a number of dimensions, such as input requirements, CPU time, and accuracy for a given geographic region. Different choices for inputs and operators yield data products that can conform to the user's basic requirements, but differ substantially in terms of various aspects. The user does not have all the information needed to recognize which final data products are feasible, but she definitely has important preferences about the properties of these end products. Thus, rather than specifying a goal,

[*]QSS Group Inc.
[†]QSS Group Inc.

[1]The term *coregistered* denotes the fact that corresponding pixels describe the same point in space, meaning the images describe the same region with an identical resolution and projection.

she needs to specify goal preferences and goal constraints. IMAGEbot must attempt to find and generate the most preferred data product given this specification.

In this paper, we discuss the data production problem, and more specifically, the use of planning to address the data production problem and the use of preferences to bias the search for a plan. Our work provides an interesting and important application domain for preference reasoning, as well as an interesting example of the use of preferences to guide search. Currently, IMAGEbot supports only simple, unconditional preferences, and one of our goals is to explain some of the challenges that we see in integrating more powerful and useful preference reasoning techniques.

2 The Data Production Problem

We applied IMAGEbot to the domain of the Terrestrial Observation and Prediction System (TOPS). In this section we explain the data production problem in this domain in more detail. Later sections discuss the view of data integration as planning and the algorithms used to implement it.

2.1 TOPS

The Terrestrial Observation and Prediction System (TOPS, http://ecocast.arc.nasa.gov) is an ecological forecasting system that assimilates data from Earth-orbiting satellites and ground weather stations to model and forecast conditions on the surface, such as soil moisture, vegetation growth and plant stress [Nemani *et al.*, 2002]. TOPS customers include scientists, farmers and land managers. With such a variety of customers and data sources, there is a strong need for a flexible mechanism for producing the desired data products for the customers, taking into account the information needs of the customer, data availability, deadlines, resource usage (some models take many hours to execute) and constraints based on context (a scientist with a palmtop in the field has different display requirements than when sitting at a desk). IMAGEbot provides such a mechanism, accepting goals in the form of descriptions of the desired data products.

The goal of TOPS is to monitor and predict changes in key environmental variables. Early warnings of potential changes in these variables, such as soil moisture, snow pack, and primary production could enhance our ability to make better socio-economic decisions relating to natural resource management and food production [Nemani *et al.*, 2000]. The accuracy of such warnings depends on how well the past, present and future conditions of the ecosystem are characterized.

2.2 Data Choices

There are many satellites providing observations of the Earth's surface, and for any given variable, there are generally several choices available that could provide it. However, these choices are not all equally good, and which ones are better may depend on the application. Even for a fixed application, having access to multiple data sources can provide needed redundancy

The inputs needed by TOPS include: Fractional Photosynthetically Active Radiation (FPAR) and Leaf Area Index (LAI); Temperatures (minimum, maximum and daylight average); Precipitation; Solar Radiation; Humidity. We have several potential candidate data sources at the beginning of each model run. The basic properties of the inputs are listed in Table 1. The specific data inputs that are selected will depend on goal constraints, such as requirements on resolution, coverage or resource limits, and preferences.

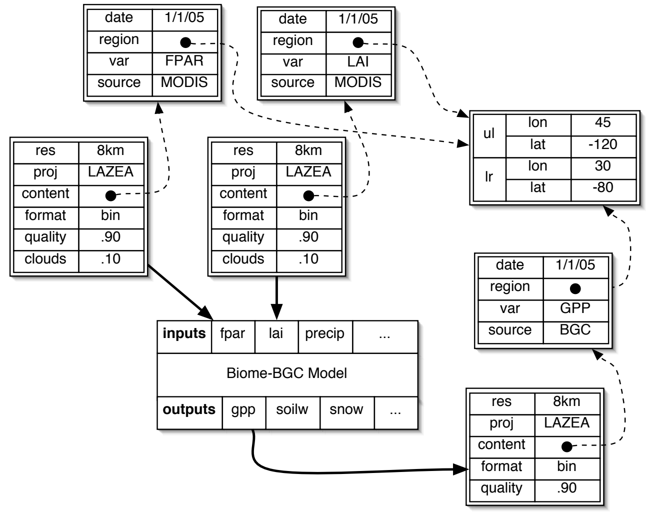

Figure 1: Structured inputs and outputs to a TOPS model

In addition to the attributes listed in Table 1, data sources also vary in terms of quality and availability – some inputs are not always available even though they should be. For example, both the Terra and Aqua satellites have experienced technical difficulties and data dropouts over periods ranging from a few hours to several weeks. Depending on the data source, different processing steps are needed to get the data into a common format. We have to convert the point data (CPC and Snotel) to grid data, and we must reproject grid data into a common projection, subset the dataset from its original spatial extent and populate the input grid used by the model. The data are then run through the TOPS model, which generates desired outputs. Figure 1 describes some of the structured inputs and outputs in a TOPS model.

2.3 Algorithm Choices

In addition to input choices, we also have several choices of models to use with the data. As with the data, the models produce results of various quality, resolution, and geographic extent. Moreover, there may sometimes be significant tradeoffs in performance versus precision. An FPAR/LAI algorithm provides a good example of this tradeoff. We can produce an FPAR/LAI pixel using either a lookup table or a radiative transfer method [Knyazikhin *et al.*, 1999]. In the case of a lookup table, we derive a Normalized Difference Vegetation Index (NDVI) from two surface reflectance channels by a means of a simple equation, and then use the NDVI value together with its landcover value as a key into a static lookup table that will give us the FPAR and LAI values. The complexity of this algorithm is O(1). On the other hand, we can use the radiative transfer method, which contains a large number of intermediate computations and has complexity $O(n \log n)$, where n is the number of pixels in the input data. This fact, together with the number of runs we may attempt, translates into a substantial difference in user time, and while the radia-

Source	Variables	Frequency	Resolution	Coverage
Terra-MODIS	FPAR/LAI	1 day	1km, 500m, 250m	global
Aqua-MODIS	FPAR/LAI	1 day	1km, 500m, 250m	global
AVHRR	FPAR/LAI	10 day	1km	global
SeaWIFS	FPAR/LAI	1 day	1km x 4km	global
DAO	temp, precip, rad, humidity	1 day	1.25 deg x 1.0 deg	global
RUC2	temp, precip, rad, humidity	1 hour	40 km	USA
CPC	temp, precip	1 day	point data	USA
Snotel	temp, precip	1 day	point data	USA
GCIP	radiation	1 day	1/2 deg	continental
NEXRAD	precipitation	1 day	4 km	USA

Table 1: TOPS input data choices

tive transfer method provides us with good results, it is not suitable for more interactive or first-pass applications, where the lookup table is sufficient. In these first-pass applications, we are looking for large abnormalities and deviations from long term normals, so high precision runs do not necessarily provide us with better results.

3 Data Production as a Planning Problem

Data processing has traditionally been automated by writing shell scripts. There are some situations when scripts are the best approach: namely, when the same procedure is to be applied repeatedly on different inputs, the environment is fairly stable and there are few choices to be made. However, in many applications, including TOPS, none of these assumptions holds. There are many different data products we would like the system to produce, there are many inputs and data-processing operations to choose from in producing those products, and the availability of these inputs can change over time. Additionally, the domain lends itself to planner-based automation; it has precisely characterized inputs and outputs and operations whose effects can also be precisely characterized. A planning approach introduces greater flexibility into the entire data-processing system. For example, rather than specifying specific data sources and formats to ensure that all the model inputs are consistent, we simply specify a constraint that all the model inputs need to be co-registered. Which inputs are chosen can then be determined based user goals and preferences and data availability.

3.1 Data Products

Data products are complex data structures. For example, a satellite image is a collection of pixels, each of which corresponds in some way to the light reflected from the Earth at a particular place and time. Viewed more abstractly, the entire image may be described in terms of the Earth system variable (or wavelength) represented, the time the image was captured, the projection, resolution and region of the image, the satellite and instrument that performed the capture, the number of "good" (e.g., not cloudy) pixels, and other attributes. In general, the structure of a data product can be described in terms of a type, such as image or animation, and a fixed set of attributes that are determined by its type. For notational convenience, we can view the object type as another attribute, and represent the object as a tuple of attribute values. For example, a file with pathname "/dir/README" and size 56 that is readable but not writable might be represented as ⟨File, "/dir/README", 56, True, False⟩. Attribute values may be primitive values, such as numbers or strings, or may themselves be structured objects or sets of structured objects.

3.2 Data Product Transforms

Data products differ from physical objects in one key aspect: they are arbitrarily replicable. If the same file is needed by multiple processes, even concurrent processes, there is no conflict beyond some disk and network contention, which can be mitigated by replication. The ease of copying data, and the relatively low cost of storage, allows us to adopt a strategy of treating data as immutable; any process that needs to modify a given file can modify a copy of the file instead. Of course, there are applications, such as databases, where destructive modification of shared data is desired, but those are not the applications this work addresses.

A DP transform is a process that takes one or more data products as inputs and produces one or more data products as outputs, such as a program that composes a number of images into a movie. Let $\mathcal{D}$ be the set of all possible data products. Formally, we can define a DP transform as a tuple $\langle I, O, \Pi, \mathcal{E} \rangle$, where $I \subset \mathcal{D}$ is a set of input data products, $O \subset \mathcal{D}$ is a set of output data products, Π is a precondition specification describing conditions on I, and $\mathcal{E}$ is a postcondition specification describing conditions on O. DP transforms are nondestructive; the data products in I are unchanged and outputs in O are newly created objects.

For example, consider the action Convert, shown graphically in Figure 2. This action takes a binary image file and converts it to a specified format (designated by the parameter `fmt`, which has the specified value "JPG" in the figure). I and O each consist of a single object, represented by the partitioned boxes in the figure. Π consists of two preconditions: that the format of the input is "binary" and the pathname is `inPath`. $\mathcal{E}$ consists of three postconditions: that the format of the output is equal to fmt, that the pathname is outPath and that all other conditions are the same as those of the input. The later condition is called a "copyof" condition [Golden, 2002]. Specifying that a given output is a "copyof" a given input provides essentially the same advantage (and has a similar interpretation) as the Strips assumption in planning: it allows us to avoid listing all the ways that the DP transform *doesn't* change the data.

A *data production problem* (DPP) is a tuple $\langle I_D, \mathcal{A}_D, \mathcal{G}_D \rangle$, where I_D is a set of initial data products, $\mathcal{G}_D$ is a specification of a set of required data products, and $\mathcal{A}_D \subseteq 2^{\mathcal{D}} \times 2^{\mathcal{D}}$ is a set of data transforms, each mapping a set of input data products to a set of output data products. Note the similarity to state-based planning problems. Instead of having one initial state and a goal state, we have multiple initial and goal data products, and actions, instead of mapping one input state to one output state, map one or more input data products to one or more output data products. Whereas a plan in classical state-based planning is a single path from the initial state to the goal state (along the graph induced by the actions), a data-production plan is a directed acyclic graph.

3.3 Large dynamic universes

Most planners make the closed-world assumption and, further, rely on grounded representations in which all actions and predicates are instantiated with all possible constants. In DP domains, as in information integration and software agent domains [Golden, 1998; Etzioni & Weld, 1994], it is impossible to identify in advance all objects in the universe. The number of available files is huge and increasing on an hourly basis. Furthermore, most actions create new objects, so the universe is not even static within the planning horizon. An examination of the standard benchmark planning problems reveals that even the hard problems typically have fewer than 100 objects total. In contrast, if we consider a single product from a single instrument (MODIS) on a single satellite (say, Terra) for a single day, there are 288 tiles. To produce a given data product, we may need to consider multiple products from multiple instruments, residing on multiple satellites, and multiple days' worth of data.

While the details of the specific files to process could be abstracted away in some cases, such an approach is not robust. Particular files may need special processing that other files do not. Sometimes needed files are missing, and substitutes from other sources must be obtained. Even worse, files are not the smallest unit of granularity; they have substructure. For example, image-processing actions act on pixels in the image — either all pixels or a subset determined by some selection criteria. Again, this detail can sometimes be abstracted away, but not always. Additionally, many actions take numeric and string arguments. Appropriate values for these arguments may be determined through constraint reasoning, but there is no way to list all possible values *a priori*. The sheer volume of possible actions makes a grounded representation unsuitable.

4 Data Product Goals and Preferences

A data product goal is a specification of one or more data products. For example, suppose a user requests an image of any resolution, in TIFF format, representing the Gross Primary Production (GPP) for the continental US on Jan 12, 2005, from any source. This goal could be represented as

$$g = \langle \text{Image}, \text{-}, \text{-}, \langle \text{Variable}, \langle \text{Date}, 2005, 12 \rangle, USA, GPP \rangle, TIFF, \text{-} \rangle$$

where "-" means "don't care." In practice, a user is not likely to be indifferent to the resolution of the image, but unless she has very specific requirements, she may not want to specify an exact value for the resolution, in case data products of that resolution are unavailable. She may, however, have constraints on the resolution. For example, anything above 8km is too coarse, and anything below 250m is too large:

$$g = \langle \text{Image}, r, \ldots, \text{-} \rangle \wedge 250\text{m} \leq r \leq 8000\text{m}$$

On the other hand, the user may want the highest (or lowest) resolution available, whatever that is. This can be specified using maximize (or minimize, respectively):

$$g = \langle \text{Image}, r, \ldots, \text{-} \rangle \wedge \text{minimize}(r)$$

What about preferences that don't involve numeric quantities? The user may have a preference of instruments used to capture the data, which are not reflected by a numeric value, such as quality or resolution. We allow the specification of an arbitrary order over values using the keyword prefer:

$$g = \langle \ldots \langle \text{Variable}, \langle \text{Date}, 2005, 12 \rangle, USA, FPAR, i \rangle \ldots \rangle \wedge \text{prefer}(i, \{\text{MODIS, AVHRR, SeaWIFS}\})$$

So far, we have only considered the case where the user has a preference over a single attribute. In practice, a user is likely to have preferences over multiple attributes, for example, wanting the best possible resolution and quality in the least possible time and at the lowest cost. Satisfying all these preferences simultaneously may not be possible, so we force the user to choose which is the most important. For example,

$$\text{prefer}(i, \{\text{MODIS, SeaWIFS}\}) \,;\, \text{minimize}(r)$$

is interpreted to mean "find the best data source first and the best resolution for that data source," whereas

$$\text{minimize}(r) \,;\, \text{prefer}(i, \{\text{MODIS, SeaWIFS}\})$$

means "give me the best resolution and the best data source that provides that resolution." It is not always possible to order preferences in this way. For example, suppose one of the attributes is cost. A user may be willing to pay more for high-quality data, but will still want a good value. It is possible to specify more complex preferences, combining two or more attributes, by using constraints. For example, to maximize data quality, q, while minimizing price, p, we could write:

$$m = \frac{q}{p+\varepsilon} \wedge \text{maximize}(m).$$

However, there are three problems with this approach:

1. A typical user is not going to be prepared to specify how to combine various quantities, such as resolution and quality, into a single preference function.
2. More complex preference functions may make the search problem much harder.
3. It only works for preferences over numeric variables. It does not provide a way of combining preferences over, say data source and resolution.

The preferences we have shown so far apply only to the goal because they are all defined on planner variables that are referred to in the goal specification. A sophisticated user might have preferences over choices that the planner makes that aren't directly reflected in goal attributes. For example, a given data source may be preferred to all others whenever that choice is made. This can be specified using universal preferences over types. For example, to indicate that NEXRAD and RUC2 are always the preferred sources of meteorological data, and NEXRAD is preferred to RUC2, we write:

$$\text{prefer}(\text{Source}, \{\text{NEXRAD, RUC2}\}).$$

This preference gets expanded out into preferences over all variables of type Source that appear in the plan, and these preferences are all unordered with respect to each other.

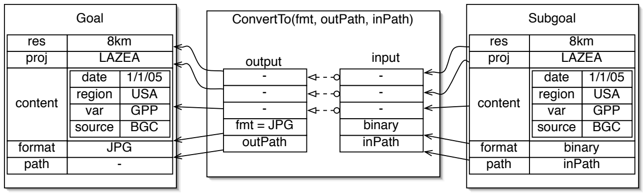

Figure 2: Goal regression for structured objects.

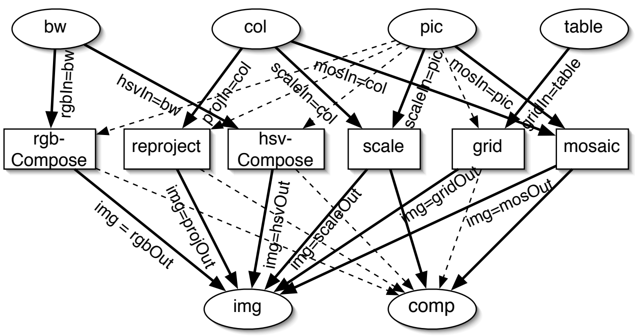

Figure 3: Lifted planning graph with constraints

5 Optimality and Search

We plan by converting the planning problem into a CSP. For most problems, the CSP's search space is infinite. Thus, the construction and solution of the CSP is incremental and proceeds backward from the goal. The CSP contains: 1) Boolean variables for all causal links, actions and conditions. 2) Variables for all parameters, input and output variables and function values. 3) For every condition in the graph, a constraint specifying when that condition holds (for conditions supported by causal links, this is just the XOR of the link variables). 4) For conjunctive and disjunctive expressions, a constraint that represents the conjunction or disjunction of the boolean variables corresponding to appropriate sub-expressions. 5) For every arc (causal link) in the graph, constraints specifying the conditions under which the supported fluents will be achieved (i.e., $\gamma_p^\alpha \Rightarrow p$, where γ_p^α is the precondition of needed to achieve p). 6) User-specified constraints. 7) Constraints representing structured objects.

In order to find an optimal plan, the planner needs to ensure that plans that assign preferred values to preferred variables are considered before plans that do not. We achieve this by the simple expedient of ordering preferred variables first, in the order of preference. When there are multiple unordered preferred variables, such as those generated by type preferences, they are ordered arbitrarily. The resulting plan is guaranteed to be Pareto optimal.

Our approach to achieving optimality can have a very bad effect on planner performance. The planner is forced to immediately search over variables that would not normally be considered until much later. These variables may have large domains that would have been pruned considerably by propagation if considered later, and the impact of infeasible value choices may likewise not be apparent until later in the search. Even worse, the domain of a variable may be infinite, in which case it could be impossible for our algorithm to find a plan in finite time. We have a number of tricks to mitigate these factors:

Planning graph propagation: At the beginning of search, the planner constructs a *lifted planning graph* representation of the search space. Whereas a conventional planning graph [Blum & Furst, 1997] is a grounded representation, consisting of ground actions and propositions, a *lifted* planning graph contains variables. Whereas a standard planning graph has only mutual exclusion constraints, our lifted planning graph contains constraints that specify the values of parameters in actions and conditions in terms of other parameters (Figure 3). We have developed a novel constraint propagation algorithm that exploits the structure of the planning graph to eliminate values from variable domains when those values could not appear in any possible plan. For example, resolution is specified as an integer, which has an infinite domain, but in practice, only a small number of resolutions are available. For example, the available resolutions for most data products in TOPS are limited to 1 degree, 40 km, 8 km, 1 km and 250 m, and some products are only available in two or three resolutions. This is very easy to discover using action-graph-based propagation.

Variable independence: Data processing domains have considerable parallelism, rendering many choices independent of each other. For example, running the TOPS BGC model requires 7 inputs. If we wish to do 7 TOPS runs, for example, to construct a weekly composite, we will need 49 variables. There happen to be 7 potential data sources for each of the variables. A type preference could easily result in separate preferences on all 49 variables, resulting in 7^{49} combinations to search over. However, as long as the resolution is fixed, the choice of values for one day will not in any way affect the choices for the remaining days, and inputs for a given day are also independent. Thus, instead of needing to consider 7^{49}, we can reduce that to 7×49.

6 Discussion

We provided an overview of the problem of data production, the planning-based approach we use to address it, and the important role the preferences and optimization play in this problem. There are many things we would like to improve in this system and many interesting issues that it raises. We now wish to discuss a number of them:

Preference Language Currently, users can express unconditional preferences over the values of an attribute. Semantically, we treat these preferences as *ceteris paribus* preferences, and our planner generates a data-product that is Pareto-optimal with respect to the partial order induced by these *ceteris paribus* preferences [Boutilier *et al.*, 2004]. However, our current preference specification language is restricted. First, some preferences in our domain are conditional. For example, a user may prefer the best possible resolution if the data quality is good, while preferring to save time and bandwidth if the quality is poor. Second, we would like to handle preference tradeoffs better. This has two aspects: (1) The strength of a preference may differ depending on context. Thus, we want to express the fact that sometimes it is desired to find the best data source first and then give the best resolution, and sometimes we would like to do the opposite. This naturally leads to the notion of conditional importance used in TCP-nets [Brafman & Domshlak, 2002]. (2) We would like to express more subtle tradeoffs over numeric variables. Earlier, we showed how we can define a new parameter (the ratio of data quality and price, in our example) and express a preference over it. However, in practice, we do not have a single preference for the value of this ratio. Rather, we prefer tradeoffs at different value regions.

Both conditional preferences and conditional importance relations are expressible in the language of TCP-nets. PrefPlan [Brafman & Chernyavsky, 2005] is a recent planner developed in order to allow for more flexible, preference-based, goal specification. Both PrefPlan and IMAGEBot use a constraint-based planning approach, but PrefPlan supports preferences specified in the language of TCP-nets. Thus, we believe that adapting IMAGEbot to handle such conditional preferences and importance relations should not be difficult. Handling the type of relative importance relations described in (2) above appears to be more challenging, but we are hopeful that similar algorithmic techniques can be used.

Search Efficiency It is well known that variable ordering is one of the most important factors influencing the solution time of constraint satisfaction problems. Our CSP-based planner (as well as PrefPlan's) must first instantiate variables on which preferences have been expressed. This can restrict our ability to select any variable orderings. The reason for this is simple – suppose A and B are variables such that we have preferences over A, but not over B, and that A's feasible values are a function of B's value. If we order B before A, we may first instantiate B to a value that constrains A to a less preferred value, and if that eventually leads to a feasible solution, it may not be a Pareto-optimal one. However, it may be the case that it is much easier to solve the problem when B is ordered before A. We believe that investigating ways of overcoming this problem is an important practical issue in preference-based constrained optimization. We see two possible approaches that might work but need to be worked out more carefully. The first is the derivation of induced preferences. In the above example, the preferences over A induce preferences over values of B, i.e., those values that are compatible with A's more prefered values. However, it seems that, in general, working out these dependent preferences will be quite complicated. Another possible direction is using limited backtracking or some branch-and-bound approach. In this approach, we would allow B to be ordered before A; we would attempt to heuristically induce preferences over B and use them to order B's values; and if a solution is obtained, we would then do additional limited search in order to verify its optimiality.

Problem-Focused Preference Elicitation Preference elicitation is a major concern for us. Our domains have a large number of objects, variables, and alternative operators. We would like to minimize the effort required by the user in order to specify a preference relation that is sufficiently rich to identify the optimal data product. We would like to use the lifted planning graph to generate some information about reachable data products, analyze what preference information can differentiate between these products efficiently, and ask the user for this preference information.

Another challenging aspect of our domain is the lack of certainty about certain variable values. For instance, some operations simply fetch data from data repositories. The quality of such data is not known ahead of time. Thus, if some image was taken on a day with substantial cloud cover, it may be unusable. Similarly, some of our operators involve the use of sophisticated models, the output of which is not know before the actual computation is performed. Thus, ideally, we need an interactive preference elicitation process: First, we analyze the problem instance to recognize the variables on which we are most likely to need preference information. Then, when we execute our plans and monitor intermediate results. If they are outside some expected parameters, we need to reinvolve the user in order to recognize whether an alternative plan should be pursued (and which plan).

References

[Blum & Furst, 1997] Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *J. Artificial Intelligence* 90(1–2):281–300.

[Boutilier *et al.*, 2004] Boutilier, C. Brafman, R.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional *ceteris paribus* preference statements. *J. Artificial Intelligence Research* 21.

[Brafman & Chernyavsky, 2005] Brafman, R. I., and Chernyavsky, Y. 2005. Planning with goal preferences and constraints. In *Proc. 15th Intl. Conf. AI Planning and Scheduling (ICAPS)*.

[Brafman & Domshlak, 2002] Brafman, R. I., and Domshlak, C. 2002. Introducing Variable Importance Tradeoffs into CP-Nets. In *Proc. 18th Conf. Uncertainty in Artifical Intelligence*.

[Etzioni & Weld, 1994] Etzioni, O., and Weld, D. 1994. A softbot-based interface to the Internet. *C. ACM* 37(7):72–6.

[Golden, 1998] Golden, K. 1998. Leap before you look: Information gathering in the PUCCINI planner. In *Proc. 4th Intl. Conf. AI Planning Systems*.

[Golden, 2002] Golden, K. 2002. DPADL: An action language for data processing domains. In *Proceedings of the 3rd NASA Intl. Planning and Scheduling workshop*, 28–33. to appear.

[Knyazikhin *et al.*, 1999] Knyazikhin, Y.; Glassy, J.; Privette, J. L.; Tian, Y.; Lotsch, A.; Zhang, Y.; Wang, Y.; Morisette, J. T.; Votava, P.; Myneni, R. B.; Nemani, R. R.; and Running, S. W. 1999. MODIS Leaf Area Index (LAI) and Fraction Of Photosynthetically Active Radiation Absorbed by Vegetation (FPAR) Product (MOD15) Algorithm Theoretical Basis Document. http://eospso.gsfc.nasa.gov/atbd/modistables.html.

[Nemani *et al.*, 2000] Nemani, R.; White, M.; Votava, P.; Glassy, J.; Roads, J.; and Running, S. 2000. Biospheric forecast system for natural resource management. In *Procceedings of GIS/EM -4*.

[Nemani *et al.*, 2002] Nemani, R.; Votava, P.; Roads, J.; White, M.; Thornton, P.; and Coughlan, J. 2002. Terrestrial observation and prediction system: Integration of satellite and surface weather observations with ecosystem models. In *Proceedings of the 2002 International Geoscience and Remote Sensing Symposium (IGARSS)*.